Computers & Graphics 33 (2009) 414–423

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

2D shape morphing via automatic feature matching and hierarchical interpolation

Wenwu Yang, Jieqing Feng *

State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou 310027, PR China

ARTICLE INFO

Article history:
Received 8 December 2008
Received in revised form
2 March 2009
Accepted 3 March 2009

Keywords:
Shape morphing
Animation
Feature matching
Path interpolation

ABSTRACT

The paper presents a new method to interpolate a pair of 2D shapes that are represented by piecewise linear curves. The method addresses two key problems in 2D shape morphing process: feature correspondence and path interpolation. First, a robust feature metric is defined to measure the similarity of a pair of 2D shapes in terms of visual appearance, orientation and relative size. Based on the metric, an optimal problem is defined and solved to associate the features on the source shape with the corresponding ones on the target shape. Then, a two-level hierarchical approach is proposed to solve the corresponding features interpolation trajectory problem. The algorithm decomposes the input shapes into a pair of corresponding coarse polygons and several pairs of corresponding features. Then the corresponding coarse polygons are interpolated in an as-rigid-as-possible plausible way; meanwhile the corresponding features are interpolated using the intrinsic method. Thus interior distortions of the intermediate shapes could be avoided and the feature details on the input shapes could be well preserved. Experimental results show that the method can generate smooth, natural and visually pleasing 2D shape morphing effects.

1. Introduction

Shape morphing, also known as shape blending or shape interpolation, is an important technique in computer graphics, which transforms a source shape to a destination shape smoothly. It has wide applications such as producing special effects in film or television, cartoon in-between automation [1], and surface reconstruction from contours [2] etc.

The source and destination shapes can be described by images or planar curves in 2D space [3,4], or surfaces or volumes in 3D space [5,6]. A satisfied morphing method should create the natural and smooth transition between the source and destination shapes and generate the intermediate ones that preserve the appearances and geometric properties of the input shapes. The process of morphing consists of two key problems: (1) feature correspondence problem, which is about how to establish the correspondence between the geometric features of the source and destination shapes; (2) path interpolation problem, which aims to find the trajectory that the corresponding elements follow during the morphing. To automatically establish the feature correspondence and generate natural and smooth morphing in different applications is still a challenging work. Therefore, both of the problems still draw much attention. In this work, we provide new solutions to the two problems for 2D shape morphing.

No matter a shape is in 2D (e.g., planar curve) or in 3D (e.g., mesh surface), it usually consists of features. For example, a human shape, its features include head, arms, feet, and body. The goal of feature correspondence in the morphing is to associate similar features on the input shapes as correctly as possible. In some applications, such as in 2D cartoon animation, the automatic feature specification and correspondence establishment will be important for the efficiency consideration, because there may be tens of thousands of key frames in a production, and in each pair of adjacent key frames the features on the 2D shapes may be dense for the sake of splendid visual effects. The questions arisen are how to detect the features on a shape and how to establish the feature correspondence on the input shapes as automatically as possible. In our approach, the features on a 2D shape are segmented by detecting the feature points of underlying curve. Meanwhile, a metric is defined to measure the similarity of a pair of features in terms of visual appearance, orientation and relative size. Based on the similarity metric, the similar features on the input shapes can be effectively associated by employing the dynamic programming technique.

In our solution to the path interpolation problem, a two-level hierarchical path interpolation is proposed. The approach first decomposes the input 2D shapes into a pair of coarse polygons and several pairs of corresponding features. Then the interpolation

* Corresponding author.
E-mail addresses: wwyang@cad.zju.edu.cn (W. Yang), jqfeng@cad.zju.edu.cn (J. Feng).

0097-8493/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2009.03.007

is conducted in three steps. First, the two coarse polygons are blended to generate the intermediate coarse polygon in an as-rigid-as-possible plausible way. The intermediate coarse polygon determines the location and orientation of each intermediate feature. Then, the corresponding features are blended using the intrinsic method. At last, the intermediate shape is reconstructed by attaching all the intermediate features onto the intermediate coarse polygon without gaps. In this way, the 2D shapes are smoothly and naturally blended where the shape interior distortions are effectively avoided and the details of the corresponding features are well preserved.

Our contributions:

- A robust metric that measures the similarity of a pair of features based on their visual appearances, orientations and relative size.
- A simple and efficient two-level hierarchical interpolation algorithm for the trajectory of the corresponding features.
- Practical enhancement using a Gaussian filter for morphing 2D shapes with abundant local details.

2. Related work

The problems of feature correspondence and path interpolation between two shapes have been (and are still being) studied in the morphing community [7]. Many algorithms have been introduced and we discuss here only those that are closely related to our method.

Feature correspondence: The step of feature correspondence tries to associate similar regions on the two shapes. In [4], a physically based method was introduced to associate the vertices on two polygonal shapes. In the method, it is assumed that each polygon is composed of pieces of wires, and the dynamic programming technique is employed to find the solution to the vertex correspondence, which means the minimal work required to bend and stretch the source wire to the corresponding destination wire. Zhang [8] established the vertex correspondence based on the similarity between pairs of triangles formed by the shape vertices. All these methods make use of the local geometric properties, such as vertex angle, edge length, or triangle area, to establish the correspondence of the regions between two input shapes. However, these local geometric properties are usually not enough to capture the properties of the features which are associated with shape meaningful parts. Therefore it cannot always establish satisfying feature correspondence on the input shapes.

In [9–11], the feature correspondence is established in a semi-automatic way, which allows user to specify one-to-one corresponding feature points on the input shapes. Liu et al. [12] extracted the feature points from the input shapes and established the one-to-one correspondence between them automatically. They designed a metric to measure the similarity of the features delimited by the feature points by using the shape property of the feature where the shape property is described via the principal component analysis [13]. The shape property of a feature involves the computations of the convexity and the tangent (or normal) direction of the feature. Unfortunately, there may be no definitive solution to both of them for some cases. For example, given an **S**-like feature, the upper part is convex and the bottom part is concave, so it is ambiguous about its convexity.

Other shape similarity metrics were also introduced [14,15]. In [14], the shape is represented by a set of uniformly sampled points on the shape, and two shapes' similarity is measured by their shape contexts which quantify shapes' features. For each sample point on a shape, the shape context describes the distribution of all the other points' relative positions in a spatial histogram. Mortara and Spagnuolo [15] extracted an approximate skeleton to describe the shape topology, and established the vertex correspondence of input shapes by matching their approximate skeletons.

Path interpolation: Because the straightforwardly linear interpolation of the shape boundary tends to produce the intermediate shapes with shrinkages, Sederberg et al. [16] proposed an intrinsic interpolation method, which can effectively avoid the shrinkages by interpolating the edge length and vertex angle of the input shapes. However, their method considers only the shape boundary, thus the interiors of the input shapes may be distorted during the morphing. In [9,17,18], the boundary interpolation is improved by the interpolation of compatible triangulations or star-skeletons. These methods can preserve the areas or volumes of the input shapes since they take into account the shapes' interiors. However, these methods are more suitable for simple shapes, since the generation of compatible triangulations or star-skeletons is not trivial for the self-intersecting shapes. In addition, the generation of compatible triangulations or star-skeletons is time consuming sometimes.

Multiresolution representations [19] decompose a shape into a low-resolution base and many levels of details. The low-resolution base can be considered as an approximate frame of the shape. Based on different multiresolution representations, several algorithms [20,21] are proposed to interpolate the low-resolution bases and details of the input shapes, and then to reconstruct the intermediate shape from the intermediate base and the intermediate details. These algorithms can achieve pleasing results: not only the overall shapes of the input objects are blended naturally but also the details of the input objects are well preserved. However, in the multiresolution approaches the low-resolution base may be quite different from the overall shape of the input object, thus the intermediate base shapes will influence the final interpolation quality greatly.

In the next section, we first give a brief overview of the proposed method, and then describe the algorithms for feature correspondence and path interpolation in details. After showing and discussing the experimental results in Section 4, we conclude this paper in Section 5.

3. Algorithm

The input to our algorithm are two 2D shapes: a source one and a destination one, represented by two piecewise linear curves. Our algorithm first detects the feature points on the input shapes, as shown in Fig. 1(a). These detected feature points delimit each shape into several segments and each of them usually associates with a meaningful part of the shape, which is called a "feature" in our context.

In the phrase of feature correspondence, based on the similarity of the features, our algorithm associates the features on the source and destination shapes automatically, i.e., establish the one-to-one correspondence between the feature points of the two shapes. The result is shown in Fig. 1(b). Note that some feature points on the source or(and) destination shapes are redundant and there are no corresponding counterparts on the other shape. Having a one-to-one feature points correspondence established, other vertices on the two shapes are automatically aligned based on the proportional chord-length principle. Then a two-level hierarchical interpolation algorithm is employed to compute the trajectory of the corresponding features. The interpolation algorithm is simple and efficient, and ultimately

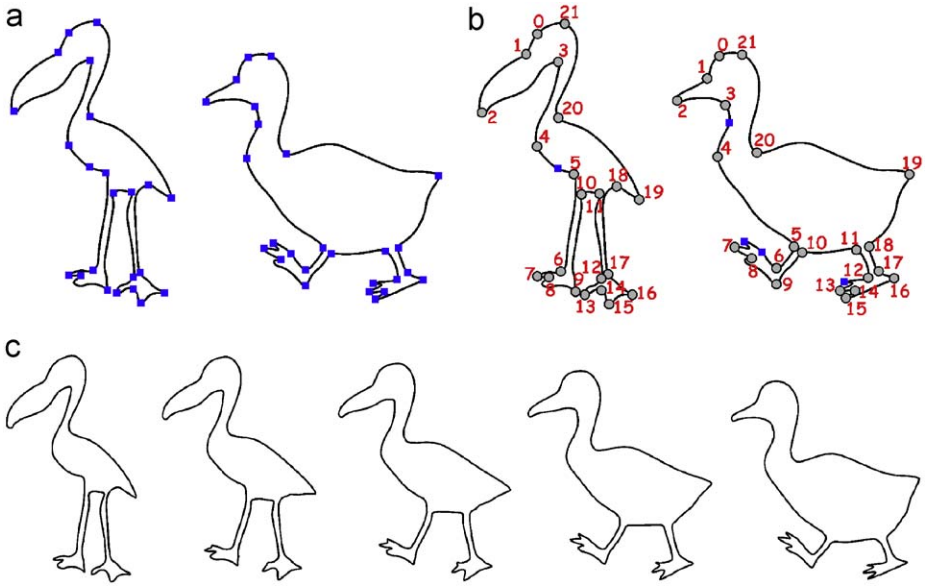

Fig. 1. Overview of the method. (a) Input shapes with the detected feature points, (b) automatic correspondence between feature points, and (c) morphing sequence.

produces the pleasing morphing effect from the source shape to the destination shape (see Fig. 1(c)).

3.1. Feature points detection

An ideal feature points detection approach can extract feature points from a shape such that the shape is segmented into several semantic components. However, the shape semantic structure usually relies on the human perception, thus it is difficult to be described only by the shape geometry information. Therefore, instead of semantic components, our method segments a shape into several meaningful features by detecting the points with high visual salience. In human perception, the cusp, inflection or curvature extrema points usually draw more attention than other points, thus it is reasonable to assign high visual salience to them. To detect these points, we use a simple and efficient algorithm [22]. This algorithm first detects the corners which are defined as the position where a triangle with specified opening angle (α_{max}) and size (d_{min}) can be inscribed in the shape, and then discards the redundant corners that are in the consecutive points. The parameters can be set as $\alpha_{max} = 165^{\circ}, d_{min} = 0.015L_s$ where L_s is the total edge length of the shape, and with the parameter configuration the feature points can be well detected, as illustrated in Fig. 1(a).

3.2. Feature correspondence

In this part, we will introduce a metric to measure the similarity of a pair of features. Based on this metric the correspondence between the feature points on the source and destination shapes can be established well by using the dynamic programming technique.

3.2.1. Feature similarity

Intuitively a shape is still the same one after it has been translated, rotated or uniformly scaled, as shown in Fig. 2(a). So a shape does not change its visual appearance under the similarity transformation which includes translation, rotation and uniform scaling or a combination of them. Here we define "visual equivalence class" of a feature as all of shapes obtained by conducting similarity transformations on the feature. Assume $f(t)$ is a line strip that represents a feature and its "visual equivalence class" is denoted as S_f, then S_f is described as follows:

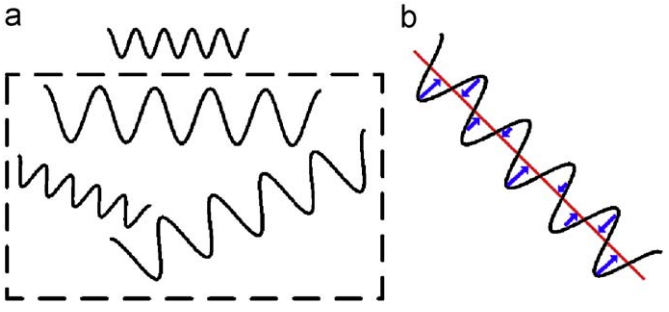

Fig. 2. (a) Top: a spring-like feature; bottom (in dashed box): several shapes in the feature's "visual equivalence class". (b) The appearance similarity of a feature represented by a line segment with the spring-like feature in (a) is measured by the average distance (as shown by the arrow) between the line segment and the closest shape obtained from the "visual equivalence class" of the spring-like feature.

$$S_f = \{\widetilde{f}(t) | \widetilde{f}(t) = f(t)A, A \in A_s\}$$

where A_s is the set of the similarity transformation matrices.

Given a pair of features $f^1(t)$ and $f^2(t)$ with m and n vertices, respectively, we first find the closest element in S_{f^1} to the feature $f^2(t)$ in L^2 norm by minimizing

$$\widetilde{A} = \arg\min_{A \in A_s} \int |f^1(t)A - f^2(t)|^2 \, dt \tag{1}$$

where the resulting similarity transformation matrix $\widetilde{A}$ determines the closest shape $f^1(t)\widetilde{A}$. Let $\mathbb{v}(f^1, f^2)$ be the visual appearance similarity between the features $f^1(t)$ and $f^2(t)$. Since the shape $f^1(t)\widetilde{A}$ has the smallest distance to the feature $f^2(t)$ among the shapes in S_{f^1}, it is reasonable to compute $\mathbb{v}(f^1, f^2)$ by using

$$\mathbb{v}(f^1, f^2) = \sqrt{\int |f^1(t)\widetilde{A} - f^2(t)|^2 \, dt} \tag{2}$$

where $\mathbb{v}(f^1, f^2)$ approximately describes the average distance of the corresponding points on the line strips $f^1(t)\widetilde{A}$ and $f^2(t)$.

An example is shown in Fig. 2(b). It is obvious that the smaller the value of $\mathbb{v}(f^1,f^2)$ is, the more similar in visual appearance the features $f^1(t)$ and $f^2(t)$ are.

In addition, we can extract rotation angle $\widetilde{A}_\theta$ and scaling ratio $\widetilde{A}_\lambda$ from the optimal similarity transformation matrix $\widetilde{A}$. $\widetilde{A}_\theta$ and $\widetilde{A}_\lambda$ express the orientation difference and the relative size of the features $f^1(t)$ and $f^2(t)$. Finally we define the total similarity: $\mathscr{D}(f^1,f^2)$ between the two features: $f^1(t)$ and $f^2(t)$, in terms of their visual appearances, orientations and relative size:

$$\mathscr{D}(f^1,f^2)=\mathbb{v}(f^1,f^2)+\mathbb{r}(f^1,f^2)+\mathbb{s}(f^1,f^2) \tag{3}$$

where $\mathbb{r}(f^1,f^2)=|\widetilde{A}_\theta|/\pi$ and

$$\mathbb{s}(f^1,f^2)=\begin{cases}1-\widetilde{A}_\lambda, & \text{if } \widetilde{A}_\lambda<1\\ 1-(\widetilde{A}_\lambda)^{-1}, & \text{otherwise}\end{cases}$$

Note that in our implementation the input shapes are normalized at first, so the value of $\mathbb{v}(f^1,f^2)$ is within the interval $[0,1]$. Meanwhile the value of $\mathbb{r}(f^1,f^2)$ or $\mathbb{s}(f^1,f^2)$ is obviously within $[0,1]$.

The next issue is how to solve Eq. (1). As mentioned above, each feature is a line strip, so the vertices on the feature can be parameterized in $[0,1]$ based on the proportional chord-length principle. Therefore, for the feature $f^1(t)$ with m vertices and the feature $f^2(t)$ with n vertices, two vertex parameter sequences can be obtained: $(t_1^1,t_2^1,\ldots,t_m^1)$ and $(t_1^2,t_2^2,\ldots,t_n^2)$, where $t_1^1=t_1^2=0$; $t_m^1=t_n^2=1$. By merging unique elements in the two parameter sequences, a new parameter sequence is generated: $(\bar{t}_1,\bar{t}_2,\ldots,\bar{t}_M)_{(M\leqslant m+n)}$. Thus the one-to-one vertex correspondence between $f^1(t)$ and $f^2(t)$ is established. Therefore, we have

$$\int|f^1(t)A-f^2(t)|^2\,dt=\sum_{i=1}^{M-1}\int_{\bar{t}_i}^{\bar{t}_{i+1}}|l_i^1(t)A-l_i^2(t)|^2\,dt \tag{4}$$

where $l_i^1(t)$ and $l_i^2(t)$ represents the i-th line segments of the refined features $f^1(t)$ and $f^2(t)$, respectively. Furthermore, after reparameterizing t as $u=(t-\bar{t}_i)(\bar{t}_{i+1}-\bar{t}_i)^{-1}$, Eq. (4) turns to be

$$\sum_{i=1}^{M-1}\int_{\bar{t}_i}^{\bar{t}_{i+1}}|l_i^1(t)A-l_i^2(t)|^2\,dt=\sum_{i=1}^{M-1}(\bar{t}_{i+1}-\bar{t}_i)\int_0^1|p_i(u)A-q_i(u)|^2\,du \tag{5}$$

where $p_i(u)$ (or $q_i(u)$) represents the i-th line segment of the refined feature $f^1(t)$ (or $f^2(t)$) and is reparameterized in interval $[0,1]$. By substituting Eq. (5) into Eq. (1), we find that Eq. (1) becomes a shape matching problem. There existed an analytic solution for 2D case [23]. Similarly, we obtain the analytic solution to Eq. (2), which is given in Appendix A.

Numeric examples: Here we will compare our feature similarity metric with that introduced by Liu et al. [12]. Liu et al. defined a covariance matrix of sample points over a feature and computed the shape property of the feature as $\varepsilon\lambda_N(\lambda_N+\lambda_T)^{-1}$, where λ_N (or λ_T) is the eigenvalue of the covariance matrix, which associates with the eigenvector that points in normal direction (or tangent direction) of the feature, and ε is 1 (or -1) when the feature is convex (or concave). Then they evaluated the similarity of visual appearances of two features as the absolute value of the difference of their shape properties. This metric is similar to our similarity metric component $\mathbb{v}(f^1,f^2)$.

However, Liu et al.'s [12] metric cannot be applied to the features with inflections. For example, an **S**-like feature, its convexity is ambiguous (i.e., the value of ε). As shown in Section 3.2.2, the combinatorial feature (i.e., a combination of two adjacent features) is adopted in our approach and it may be an **S**-like feature, thus Liu et al.'s metric is no longer applicable. Furthermore, $\mathbb{v}(f^1,f^2)$ is more elegant than Liu et al.'s metric.

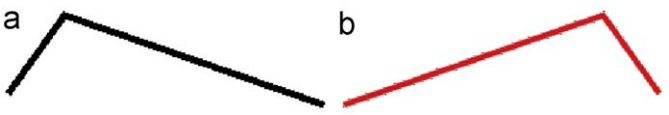

Fig. 3. Two symmetric shapes along the y axis.

For example, the shapes shown in Fig. 3(a) and (b) are symmetric along the y axis. In 2D, they are two different shapes in visual appearance, but the numeric result computed by Liu et al.'s metric is 0.0, which means the two shapes are completely similar. Our metric component $\mathbb{v}(f^1,f^2)$ gives their similarity as 0.1189, which correctly shows that there is visual appearance difference between the two shapes.

3.2.2. One-to-one correspondence

Assume there are N_1 and N_2 feature points on the source and destination shapes, which are denoted as $(S_i)_{i=1}^{N_1}$ and $(T_j)_{j=1}^{N_2}$, respectively. Obviously these feature points segment the two shapes into N_1 and N_2 features, respectively. We first define the correspondence cost of a source feature point with a destination feature point. In order to associate similar features on the source and destination shapes, it is reasonable to associate a source feature point with a destination feature point if and only if their neighboring features are similar. Therefore, the correspondence cost of a source feature point S_i with a destination feature point T_j is defined as

$$CorCost(S_i,T_j)=\sum_{p=\mathbb{v},\mathbb{r},\mathbb{s}}w_p\sum_{q=(lt,rt,lrt)}\mathscr{D}_p(f^1_{\langle i,q\rangle},f^2_{\langle j,q\rangle}) \tag{6}$$

where $f^1_{\langle i,lt\rangle}$ and $f^1_{\langle i,rt\rangle}$ are the left and right neighboring features of the feature point S_i on the source shape, respectively, and they are delimited by the feature points $(S_{i-1},\ S_i)$ and $(S_i,\ S_{i+1})$, respectively; $f_{\langle i,lrt\rangle}$ is a combinatorial feature formed by combining $f^1_{\langle i,lt\rangle}$ and $f^1_{\langle i,rt\rangle}$. The features $f^2_{\langle j,lt\rangle}$, $f^2_{\langle j,rt\rangle}$, $f^2_{\langle j,lrt\rangle}$ are defined similarly for the feature point T_j on the destination shape. Meanwhile, $\mathscr{D}_p$ is the cost associated with the difference of two features in visual appearance, orientation or size (i.e., $p=\mathbb{v},\mathbb{r},\mathbb{s}$), which is defined as

$$\begin{cases}\mathscr{D}_{\mathbb{v}}(f^1_{\langle i,q\rangle},f^2_{\langle j,q\rangle})=\mathbb{v}(f^1_{\langle i,q\rangle},f^2_{\langle j,q\rangle})\\ \mathscr{D}_{\mathbb{r}}(f^1_{\langle i,q\rangle},f^2_{\langle j,q\rangle})=\mathbb{r}(f^1_{\langle i,q\rangle},f^2_{\langle j,q\rangle})\\ \mathscr{D}_{\mathbb{s}}(f^1_{\langle i,q\rangle},f^2_{\langle j,q\rangle})=\mathbb{s}(f^1_{\langle i,q\rangle},f^2_{\langle j,q\rangle})\end{cases} \tag{7}$$

and w_p is the weight factor for the corresponding cost $\mathscr{D}_p$.

Finally, the remaining work is to find a sequential correspondence between $(S_i)_{i=1}^{N_1}$ and $(T_j)_{j=1}^{N_2}$ so that the sum of the correspondence cost of each pair of the corresponding source and destination feature points is minimum among all possible sequential correspondences. The dynamic programming technique was employed to solve this problem in our implementation, whose time complexity is $O(N_1N_2\ln N_2)$ [4].

Note that there may be redundant feature points on the source shape or(and) the destination shape and there do not exist unique optimal corresponding feature points for them on the other shape, so multiple feature points on the source shape may correspond to a feature point on the destination shape or vice versa by the dynamic programming process, as illustrated in Fig. 4. For this case we only keep the corresponding pair with minimal correspondence cost (i.e., the dominant feature pairs) and ignore the redundant feature points in order that the feature points on the source and destination shapes are associated one-to-one.

3.3. Path interpolation

The proposed interpolation algorithm is hierarchical in two levels. Firstly it decomposes each shape into an approximate polygon called "frame polygon" and several features. Then it interpolates the corresponding frame polygons and the corresponding features separately. Finally, each intermediate shape is reconstructed from an intermediate frame polygon and the intermediate features. The algorithm is similar to [11], but has two substantial improvements:

- An as-rigid-as-possible plausible method is designed to interpolate the frame polygons, which is able to avoid the distortions of the shapes' interiors.
- The intrinsic method is used to blend the corresponding features with gap-free constraints.

3.3.1. Frame polygon and features

According to the corresponding feature points, the source and destination shapes can be decomposed into a pair of corresponding frame polygons and several pairs of corresponding features, as shown in Fig. 5. Note that the frame polygon can be considered as a coarse version of a 2D shape and each of the shape features is attached to one of the frame polygon edges. Each of the frame polygon edges defines a local coordinate system for the attached feature where its starting vertex is taken as the origin and the normalized vector from its starting vertex to its ending vertex is taken as an axis, and then the attached feature is transformed into this local coordinate system, as shown at the top right of Fig. 5. Obviously the local coordinate system determines the location and orientation of the feature.

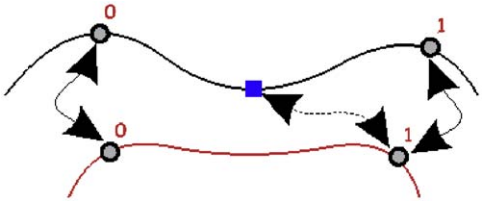

Fig. 4. There is a redundant feature point (square point) on the source shape (top). The correspondence pair of the redundant source feature point with a feature point on the destination shape (bottom) established by the dynamic programming process is ignored.

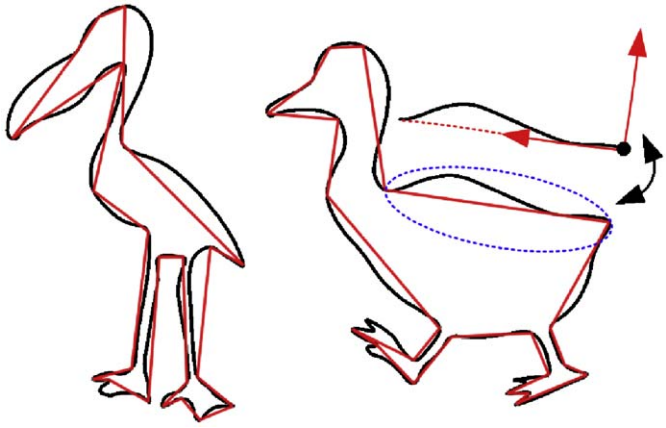

Fig. 5. Frame polygons (in red line) of the input shapes, which are generated according to the corresponding feature points shown in Fig. 1(b). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.3.2. Hierarchical interpolation algorithm

The core algorithm of the path interpolation includes three steps.

Step 1: *Frame polygon interpolation*. Since the frame polygon is the coarse version of the shape and each of its edges determines the location and orientation of the corresponding feature, it is reasonable to interpolate the source and destination frame polygons in an as-rigid-as-possible way by taking the interiors into account, in order to preserve the shape areas and change the orientations of the features linearly. However, the frame polygons of the source and destination shapes may not be simple polygons, thus the as-rigid-as-possible method which interpolates the compatible triangulations [9] cannot be employed straightforwardly. Therefore we proposed an as-rigid-as-possible plausible method for the frame polygons interpolation, which is not confined to simple polygons.

On the source frame polygon or the destination frame polygon, the method constructs a triangle at each polygon vertex with its left and right neighbor vertices, which is shown in the dashed blue lines in Fig. 6. Obviously each triangle from the source frame polygon corresponds to one from the destination frame polygon. Similar to [9], each pair of the corresponding triangles defines an optimal affine transformation. The optimal affine transformations will be interpolated in the as-rigid-as-possible way, and thus a set of intermediate affine transformations can be obtained. The intermediate affine transformations can determine the intermediate triangles. However, each frame polygon vertex belongs to three triangles and its intermediate position may not be determined by the three intermediate triangles uniquely. As [9], we define and solve an optimization problem with the intermediate affine transformations, and finally determine the intermediate positions of the frame polygon vertices uniquely. Because the triangles on a frame polygon are usually few and are overlapped each other, they will seem to be glued together and tend not to be distorted during the as-rigid-as-possible interpolation. Therefore, the intermediate frame polygons with area preservation are able to be obtained, as shown in Fig. 6, where the orientation of each edge also changes naturally.

Step 2: *Feature interpolation*. Given a source feature f^1 and the corresponding destination feature f^2 with M pairs of corresponding vertices. We translate them into the intrinsic parameters representation, i.e., edge length and vertex angle, in order that the feature orientation and size can be interpolated naturally during

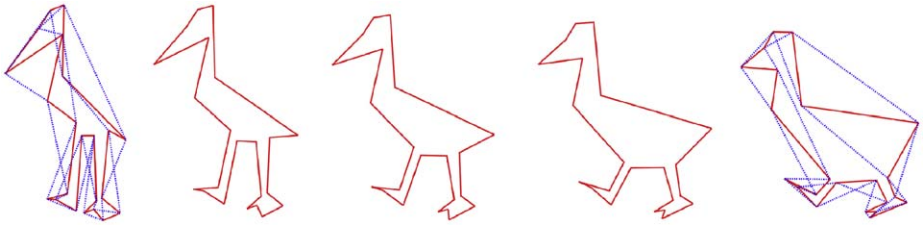

Fig. 6. Frame polygons interpolation. The leftmost and the rightmost are source and destination frame polygons, respectively.

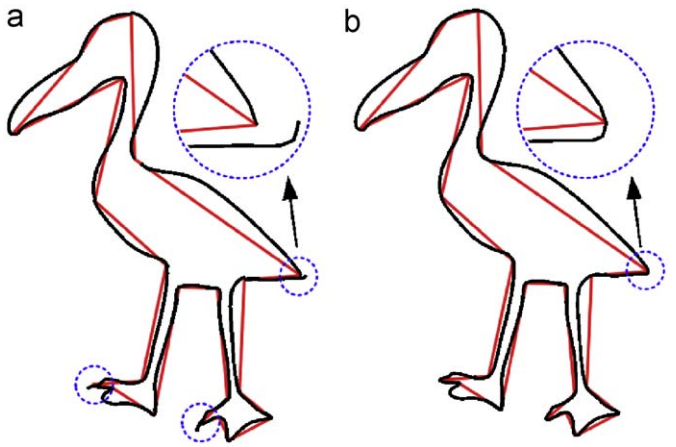

Fig. 7. Intermediate features are attached onto the intermediate frame polygon: (a) with gaps and (b) without gaps.

the morphing. Let Φ^1 be the intrinsic parameters set of the source feature f^1, which includes [16] edge lengths, angle from the angle line to the first edge, and vertex angles. In our implementation, the x axis of the local coordinate system associated with the feature is taken as the angle line. Similarly, we can define Φ^2 for f^2.

At an intermediate time t, the intermediate feature represented by the intrinsic parameter Φ^t is obtained by interpolating the corresponding intrinsic parameters in the sets Φ^1 and Φ^2:

$$\Phi^t = (1-t)\Phi^1 + t\Phi^2 \tag{8}$$

Step 3: *Reconstruction*. Now we merge the intermediate frame polygon and the intermediate features together to form an intermediate shape, as shown in Fig. 7. Because the size of each intermediate feature is usually not as same as the corresponding edge length of the intermediate frame polygon, the simple merging scheme will introduce gaps between them (Fig. 7(a)). To eliminate the gaps, as shown in Fig. 7(b), we adjust the edge lengths of the intrinsic parameters of the intermediate feature as follows. Given an intermediate feature that is represented in terms of the intrinsic parameters and its corresponding edge $\overrightarrow{v_a v_b}$ on the intermediate frame polygon, the edge lengths of the intrinsic parameters are then adjusted in the similar way of [16], such that the end vertex of the intermediate feature is coincident with the end vertex of the edge $\overrightarrow{v_a v_b}$, i.e., the vertex $(L_{\overrightarrow{v_a v_b}}, 0)$, where $L_{\overrightarrow{v_a v_b}}$ is the length of the edge $\overrightarrow{v_a v_b}$.

Then the location of the intermediate feature with the adjusted intrinsic parameters is determined by fixing its first point at the origin of the corresponding local coordinate system derived from the edge $\overrightarrow{v_a v_b}$, and finally the intermediate feature in the local coordinate system is transformed back to the global coordinate system.

4. Results and discussion

The method has been implemented in a single thread way on a PC with 2.4 GHz Intel Core 2 Duo CPU and 2 GB RAM. Table 1 shows time statistics to establish feature correspondence using various methods. The feature correspondence algorithm of our method obviously runs faster than those that conduct the correspondence on the shape vertices directly [4,8]. That is because the number of feature points on a shape is much less than the number of shape vertices in general, thus the runtime of the dynamic programming process decreases greatly in our method. Compared with the algorithm [12] which also conducts the correspondence on the feature points only, the efficiency of our algorithm drops in some extent, as shown in Table 1. The reason lies in that the computation of similarity $\mathscr{D}(f^1, f^2)$ between two features $f^1(t)$ and $f^2(t)$ is more complex than [12], which needs to solve Eq. (1) and involves the merging of two parameter sequences. However, our correspondence algorithm still achieves satisfying respondence speed. If the memory size is permitted, it is possible to improve the efficiency further by storing and reusing the already merged parameter sequences during the dynamic programming process. The interpolation algorithm of our method includes three steps. Since the as-rigid-as-possible plausible interpolation does not need the compatible triangulations and the number of triangle pairs (equivalent to the number of feature points) input to it is small, step 1 runs fast. Meanwhile, the modified intrinsic method does not have complex computations so steps 2 and 3 are also quick. In general our interpolation algorithm can generate over 300 intermediate shapes per second.

Table 1
Performance statistics for feature correspondences of input shapes (the runtime is measured in millisecond).

Fig.	VER 1 NUM	VER 2 NUM	T1 (ms)	T2 (ms)	T3 (ms)	T4 (ms)
1(a)	235	215	3752	5625	22	172
8	271	267	6140	7982	14	204
9	61	94	168	224	2	6

VER 1/2 NUM: number of vertices on the source/destination shape; T1: runtime of physical based approach by Sederberg et al. [4]; T2: runtime of fuzzy approach by Zhang [8]; T3: runtime of perceptually based approach by Liu et al. [12]; T4: runtime of our approach.

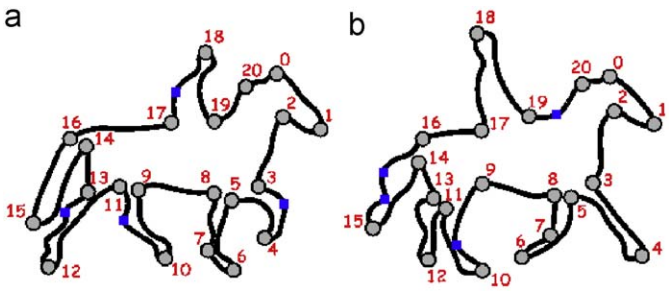

Fig. 8. The feature points on two horsemen are detected and associated by using our method.

The proposed correspondence algorithm has been applied to the shapes shown in Figs. 1(a), 8 and 9(a) and (b). The weight factors $w_{\mathbb{V}}$, $w_{\mathbb{T}}$ and $w_{\mathbb{S}}$ control the similarity degree between the corresponding source and destination features in terms of their visual appearances, orientations and relative size, respectively. In our examples, $(w_{\mathbb{V}}, w_{\mathbb{T}}, w_{\mathbb{S}})$ are (1.0, 0, 0) shown in Fig. 1(a), (1.0, 0, 0.1) shown in Fig. 8, (1.0, 0, 0) shown in Fig. 9(a) and (1.0, 0.8, 0) in Fig. 9(b). In Fig. 9(a), the similar features are associated with $w_{\mathbb{T}} = 0.0$, which means the feature orientation similarity is not considered. Therefore, although the features delimited by the feature points 0 and 1 on the input shapes are different in orientations (about 90° angle), they are still associated. During morphing the feature between 0 and 1 on the source shape (top) will rotate to the corresponding feature on the destination shape (bottom), as shown in Fig. 11(a). If the rotation effect is undesirable, we can avoid the rotation with the increased $w_{\mathbb{T}}$, which guarantees that only the features that have similar orientations are associated. For example, in Fig. 9(b) the $w_{\mathbb{T}}$ is increased to 0.8 and the feature between feature points 0 and 1 on the source shape is associated with a new feature on the

destination shape where their orientations are nearly approached. Then the rotation effect is effectively avoided, as shown in Fig. 11(b). From the given examples, we can draw that our correspondence algorithm can generate comparable results to those generated by the previous algorithms [4,8,12]. Furthermore, our correspondence algorithm sometimes works better than the previous ones, as shown in Fig. 10.

The morphing results generated by our two-level hierarchical interpolation algorithm are shown in Figs. 1(c), 11 and 12(e). As shown in Figs. 12(e) and (f), the result generated by our two-level hierarchical interpolation is comparable to that generated by the as-rigid-as-possible interpolation [9]. Because the vertex numbers in the corresponding source and destination shapes of Fig. 12(f) is large (1187 vertices), thus the as-rigid-as-possible interpolation is not efficient (i.e., 7126 ms for compatible triangulations and 22 frames/s); however, our interpolation algorithm does not need compatible triangulations and runs fast (i.e., 377 frames/s). In Fig. 13, the morphing sequences generated by the intrinsic approach [16], the feature-decomposition based approach [11] and multiresolution approach [21] are also shown. Compared with the morphing sequence generated by our approach in Fig. 1(c), their results are slightly distorted in the shape interiors and the results of feature-decomposition based approach [11] even contain the undesired extrusions which are similar to the results with gaps shown in Fig. 7(a). The reasons for the shape interior distortions shown in Fig. 13 are (1) the intrinsic approach [16] does not take the shape interior into account; (2) although

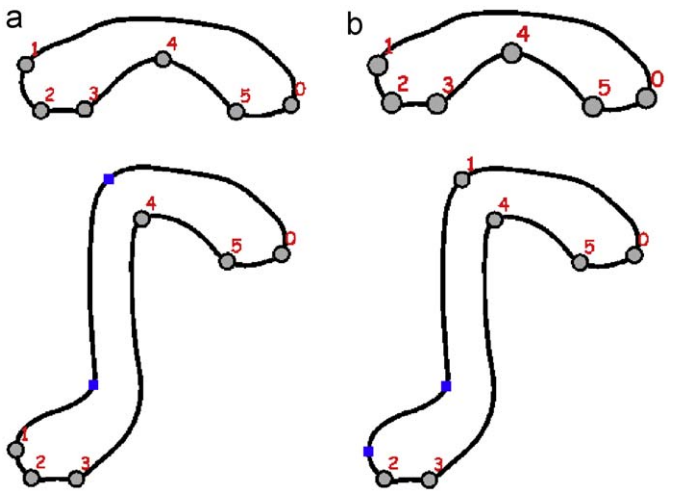

Fig. 9. The feature points on the two shapes are associated with different orientation weight factor w_r. (a) $w_r = 0.0$. (b) $w_r = 0.8$.

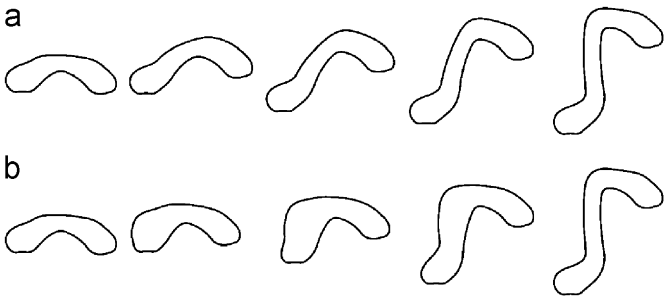

Fig. 11. Morphing sequences for two shapes with different feature points correspondence shown in Fig. 9 by using our hierarchical interpolation algorithm. (a) Correspondence is shown in Fig. 9(a) and (b) correspondence is shown in Fig. 9(b).

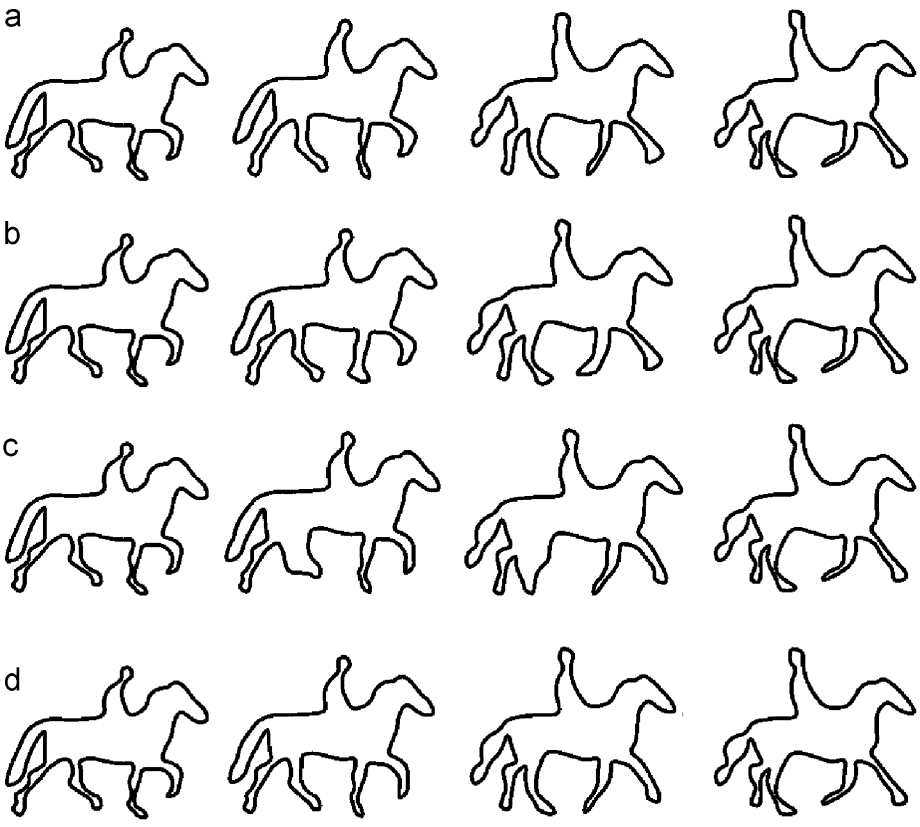

Fig. 10. Comparative results of various correspondence algorithms: (a) algorithm of [4]; (b) algorithm of [8]; (c) algorithm of [12]; (d) ours. For the comparison fairness, the third-party interpolation algorithm [16] is employed to get the morphing sequences.

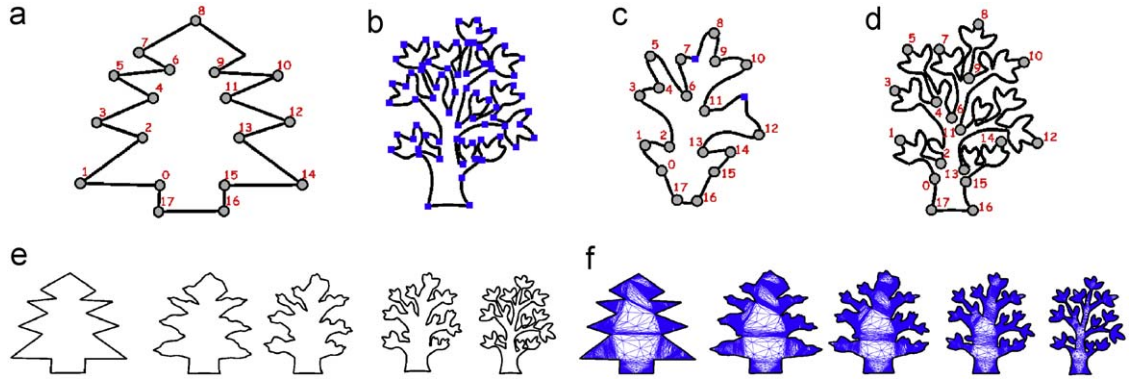

Fig. 12. (a) A source tree. (b) A destination tree with many local details. Thus a lot of feature points are detected. The two trees are depicted from [21]. (c) The smooth shape of the destination tree where only the feature points that are located on the dominant features are detected. (d) The feature points on the source tree (a) and the smooth shape (c) are associated with ($w_v = 1.0$, $w_r = 1.0$, $w_s = 0.3$). Then the corresponding feature points on (c) are transferred to the destination tree. (e) Morphing sequence using our hierarchical interpolation algorithm. (f) Morphing sequence with the compatible triangulations by using the as-rigid-as-possible method [9].

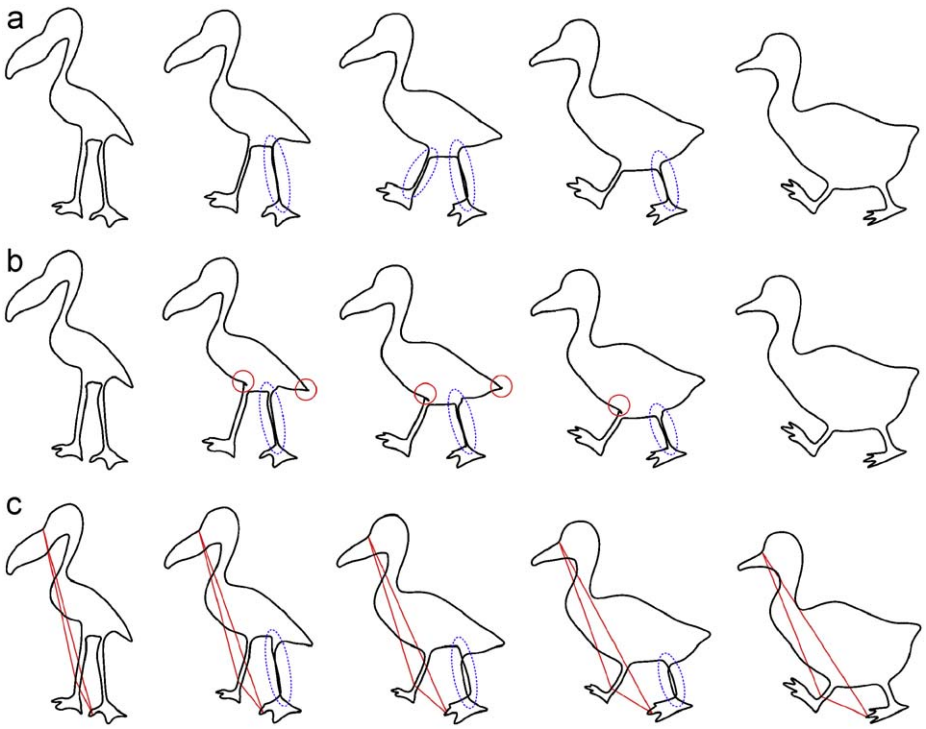

Fig. 13. Comparative results of interpolation algorithms. The correspondence of the shapes is shown in Fig. 1(b). (a) Algorithm of [16]; (b) algorithm of [11]; (c) algorithm of [21], where the blending of the low-resolution base polygons (i.e., the triangles) is shown in the red line, and note that additional vertices are inserted in the source and destination shapes to make their vertex numbers be $3 \cdot 2^n$, such that each of input shapes can be decomposed into a triangle base; Our results are shown in Fig. 1(c). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

feature-decomposition based approach [11] takes the shape interior into account, it does not interpolate the shape interior in an as-rigid-as-possible plausible way; (3) in multiresolution approach [21], the shape interior is taken into account by interpolating the low-resolution base polygons in the as-rigid-as-possible way; however, the low-resolution base polygons (i.e., the triangles) shown in Fig. 13(c) are quite different from the overall shapes of the source and destination objects, such that the interior areas of the input shapes could not be effectively preserved. In our hierarchical interpolation, the shape interior distortions are effectively avoided by interpolating the corresponding frame polygons in an as-rigid-as-possible plausible way, since the frame polygons approximate well the overall shapes of input objects.

Morphing of shapes with abundant local details: In the practical morphing application, we sometimes have to deal with the shapes with many local details. For these shapes, a lot of feature points will be detected by using the feature points detection process introduced in Section 3.1, as shown in Fig. 12(b), which will make our feature correspondence algorithm fail to associate similar features on the input shapes. To address this problem, we employ a Gaussian filter to smooth out the high-frequency local details of the shape while preserving the low-frequency global features. An example of the smoothed shape is shown in Fig. 12(c). In our

Fig. 14. The two trees have similar semantic structures, but several features that are similar in semantic level are quite different in geometry details. Thus some feature points (e.g., feature points 15) are not matched correctly using our correspondence algorithm with ($w_v = 0.8$, $w_r = 0.4$, $w_s = 0.6$), which leads to unpleasing morphing effect. Note that in [11] the pleasing morphing result is shown for the same input shapes. That is because the correspondence in [11] is correctly specified by user, and under the correct correspondence our two-level hierarchical interpolation algorithm can also achieve pleasing result.

implementation the default radius of the filter function window is taken as the average edge length of the shape to be smoothed and user can adjust it interactively. When the scales of the shape details are varying, the Gaussian filter with interactively tuning radius could be employed, i.e., large radius for the regions with fine details, while small radius for the smooth regions. Then we detect the feature points and establish the feature point correspondence on the smooth shape. At last, the corresponding feature points are mapped back to the original shape. As shown in Fig. 12, for the shape in Fig. 12(b), the feature points are detected on its smooth shape (Fig. 12(c)), and then the correspondence of feature points on the shape in Fig. 12(a) and the feature points on the smooth shape (Fig. 12(c)) is established. At last the feature points on the smooth shape (Fig. 12(c)) are mapped back to the original shape (Fig. 12(d)), which ultimately establish the feature points correspondence on the shapes in Figs. 12(a) and (d).

Limitations: Our correspondence algorithm cannot handle well the cases where the features on the input shapes are similar in semantic level but are quite different in geometry level. An example is shown in Fig. 14.

5. Conclusion

In this paper, we have proposed a robust feature correspondence algorithm and a simple and efficient two-level hierarchical interpolation algorithm. The proposed correspondence algorithm tends to associate the similar features on the input shapes. Meanwhile, the visually pleasing morphing effects are achieved by the two-level hierarchical interpolation algorithm where the shape interior distortions are effectively avoided and the feature details are well preserved.

Despite this, shape morphing is not only a technical problem but also an aesthetic problem. The evaluation of an algorithm relies on the user's perception greatly. Thus it is still important to find an objective assessment criteria. Currently we segment the shape into features by detecting the feature points (sometimes with the help of Gaussian filter), but as discussed previously the segmented features may not always be the real semantic features of the shape. Thus to detect the shape semantic features is worthy of being considered as a future work. Meanwhile, it is also a worthy work to apply the proposed approach to the morphing of complex objects, such as key frames of 2D animation, where each key frame may contain several 2D characters and each character may be composed of several shapes instead of a single one.

Acknowledgments

We would like to thank the reviewers for their valuable comments, Haihong Xia for helping with the preparation of the examples and Prof. Xiaogang Jin for the helpful discussion. This work was jointly supported by the NSF of China (60743002), the 973 Program of China (2009CB320801), the National Key Technology R&D Program (2007BAH11B02) and the NSF of Zhejiang Province (R106449).

Appendix A

Here we provide the closed-form solution to Eq. (2) by replacing $\int |f^1(t)A - f^2(t)|^2\,dt$ in Eq. (2) with Eq. (5). Let (a_i, b_i) and (c_i, d_i) be end points of the i-th line segment of the split features $f^1(t)$ and $f^2(t)$, respectively. Then Eq. (2) has the closed-form solution

$$\begin{aligned}\mathbb{V}(f^1,f^2) &= \sqrt{\int |f^1(t)\widetilde{A} - f^2(t)|^2\,dt} \\ &= \sqrt{\sum_{i=1}^{M-1}(\bar{t}_{i+1}-\bar{t}_i)\int_0^1 |p_i(u)\widetilde{A} - q_i(u)|^2\,du} \\ &= \sqrt{\sum_{i=1}^{M-1}(\bar{t}_{i+1}-\bar{t}_i)(\alpha_i+\beta_i+\frac{1}{3}\lambda_i)}\end{aligned}$$

where

$$\alpha_i = |a_i\widetilde{A} - c_i|^2$$
$$\beta_i = (a_i\widetilde{A} - c_i)(-a_i\widetilde{A} + b_i\widetilde{A} + c_i - d_i)^T$$
$$\lambda_i = |-a_i\widetilde{A} + b_i\widetilde{A} + c_i - d_i|^2$$

Appendix B. Supplementary material

Supplementary data associated with this article can be found in the online version of 10.1016/j.cag.2009.03.007.

References

[1] Fekete J-D, Bizouarn É, Cournarie É, Galas T, Taillefer F. TicTacToon: a paperless system for professional 2D animation. In: SIGGRAPH '95; 1995. p. 79–90.
[2] Kaul A, Rossignac J. Solid-interpolating deformations: construction and animation of pips. In: Proceedings of eurographics '91; 1991. p. 493–505.
[3] Beier T, Neely S. Feature-based image metamorphosis. ACM Computer Graphics 1992;26(2):35–42.
[4] Sederberg TW, Greenwood E. A physically based approach to 2D shape blending. ACM Computer Graphics 1992;26(2):25–34.
[5] Cohen-Or D, Solomovic A, Levin D. Three-dimensional distance field metamorphosis. ACM Transactions on Graphics 1998;17(2):116–41.
[6] Sheffer A, Kraevoy V. Pyramid coordinates for morphing and deformation. In: 3DPVT '04: proceedings of the 3D data processing, visualization, and transmission; 2004. p. 68–75.
[7] Wolberg G. Image morphing: a survey. The Visual Computer 1998;14(8/9):360–72.
[8] Zhang Y. A fuzzy approach to digital image warping. IEEE Computer Graphics and Applications 1996;16(4):34–41.
[9] Alexa M, Cohen-Or D, Levin D. As-rigid-as-possible shape interpolation. In: SIGGRAPH '00; 2000. p. 157–64.
[10] Carmel E, Cohen-Or D. Warp-guided object space morphing. The Visual Computer 1997;13(9/10):465–78.
[11] Yang W, Feng J, Jin X, Peng Q, Forrest AR. 2-D shape blending based on visual feature decomposition. In: CASA ,04: proceedings of computer animation and social agents 2004, Geneva, Switzerland; 2004. p. 139–46.
[12] Liu L, Wang G, Zhang B, Guo B, Shum H-Y. Perceptually based approach for planar shape morphing. In: PG '04: proceedings of the computer graphics and applications, 12th Pacific Conference; 2004. p. 111–20.
[13] Pauly M, Keiser R, Gross M, Zürich E. Multi-scale feature extraction on point-sampled models. In: Eurographics 2003; 2003.

[14] Belongie S, Malik J, Puzicha J. Shape matching and object recognition using shape contexts. IEEE Transactions on Pattern Analysis and Machine Intelligence 2002;24(4):509–22.
[15] Mortara M, Spagnuolo M. Similarity measures for blending polygonal shapes. Computers & Graphics 2001;25(1):13–27.
[16] Sederberg TW, Gao P, Wang G, Mu H. 2-D shape blending: an intrinsic solution to the vertex path problem. In: SIGGRAPH '93; 1993. p. 15–8.
[17] Shapira M, Rappoport A. Shape blending using the star-skeleton representation. IEEE Computer Graphics and Applications 1995;15(2):44–50.
[18] Surazhsky V, Gotsman C. Intrinsic morphing of compatible triangulations. International Journal of Shape Modelling 2003;9(2):191–201.
[19] Mallat SG. A theory for multiresolution signal decomposition: the wavelet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 1989;11(7):674–93.
[20] Goldstein E, Gotsman C. Polygon morphing using a multiresolution representation. In: Proceedings of graphics interface '95; 1995. p. 247–54.
[21] Hahmann S, Bonneau G-P, Caramiaux B, Cornillac M. Multiresolution morphing for planar curves. Computing 2007;79(2):197–209.
[22] Chetverikov D, Szabo Z. A simple and efficient algorithm for detection of high curvature points in planar curves. In: Proceedings of 23rd workshop of the Austrian pattern recognition groups; 1999. p. 175–84.
[23] Schaefer S, McPhail T, Warren J. Image deformation using moving least squares. ACM Transactions on Graphics 2006;25(3):533–40.